بسمه تعالی

تمرین سری سوم درس یادگیری عمیق

پوریا محمدی نسب

۴۰۰۷۲۲۱۳۸

فهرست

سوال ۱.

مقاله زیر را مطالعه بفرمایید و خلاصه‌ای از آن را در قالب یک گزارش بنویسید.

بیان مشکل:

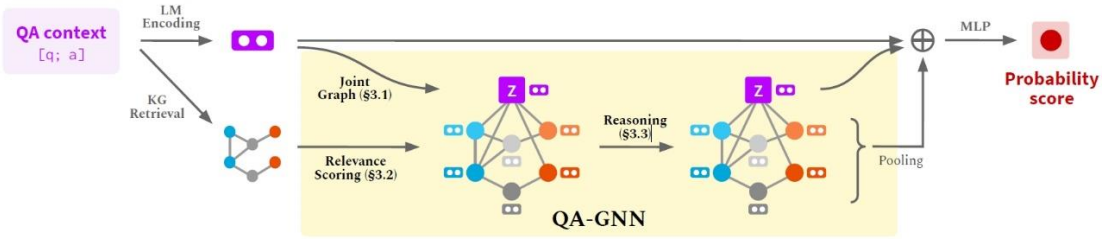
مسئله پاسخگویی(question answering) ای که از مدل های زبانی از پیش آموزش دیده(pre-trained) و گراف دانش استفاده میکنند با دو چالش اساسی در هنگام مواجهه با یک سوال و چند گزینه(QA) رو به رو هستند. اول اینکه الگوریتم باید دانش مرتبط با سوال را از بین گراف دانش بسیار بزرگ تشخیص دهد و دوم اینکه استدلالی مرتبط بین QA و گراف دانش انجام دهند.

روش پیشنهادی:

هنگامی که یک سوال و یک سری جواب انتخابی (answer choice) دریافت میشود آنها را به هم الحاق میکنیم تا یک QA به صورت [q; a] بدست بیاید. برای استنتاج یک QA مدل پیشنهادی به صورت زیر عمل میکند:

۱) در ابتدا از مدل زبانی(LM) برای ارائه QA استفاده میکند و زیر گراف G_{sub} را از گراف دانش بازیابی میکند.

۲) در این مدل QA ای که در مراحل قبل ساخته شده است به عنوان یک گره در گراف در نظر گرفته میشود و این گره به موضوعات مرتبط با آن وصل میشود و بدین ترتیب یک گراف متصل بر روی دو منبع دانش خواهیم داشت که به این گراف، اصطلاحا working graph گفته میشود که با G_w نمایش میدهیم. حال به این موضوع میپردازیم که چگونه میتوان این گراف متصل بر روی دو منبع دانش را ساخت. به طور کلی اگر دو منبع دانش را در دو سمت جداگانه قرار داد و آنها را با یک گراف در بین آنها به هم متصل کرد working graph را تشکیل داده ایم. گراف G_w یک فضای استنتاج روی گراف دانش و QA بوجود میاورد که به آن working memory نیز میگویند. ما در گراف G_w چهار نوع گره داریم. نوع اول گره های contex هستند، نوع دوم و سوم به ترتیب گره های Question و Answer هستند و نوع چهارم گره هایی هستند که از سه نوع دیگر نیستند و در شکل زیر آنها را با رنگ های بنفش، آبی، قرمز و خاکستری (به ترتیب) مشاهده میکنیم.



۳) مرحله و چالش بعدی در این مدل این است که چگونه ارتباط سازگارشونده بین گره های QA و گراف G_w را بدست بیاوریم. در این روش پیشنهادی برای هر جفت QA و نقاط معنایی گراف یک امتیاز ارتباط(relevance score) با استفاده از LM محاسبه میشود و این score های بدست آمده به عنوان یک خصیصه (feature) به هر گره در گراف اضافه میشود. اهمیت این امتیاز دهی به ارتباطات زمانی بیشتر قابل مشاهده است که درمیابیم تعداد زیادی از گره هایی که در زیر گراف دانش G_{sub} وجود دارند، بی ارتباط و زائد هستند و اگر سیستم امتیازدهی به ارتباطات وجود نداشت اثرات مخربی روی یادگیری مدل مانند overfitting و یا پیچیدگی در استنتاج داشتند. حال برای محاسبه امتیاز ارتباط از مدل زبانی از پیش آموزش دیده استفاده میکنیم به این صورت که برای هر گره text(v)

را با QA الحاق میکنیم و در واقع با استفاده از احتمالی که LM به هر text(v) میدهد آن را در رابطه زیر قرار داده و امتیاز ارتباط را محاسبه میکنیم:

$$\rho_v = f_{\text{head}}(f_{\text{enc}}([\texttt{text}(z); \texttt{text}(v)]))$$

به عبارتی دیگر امتیاز میزان ارتباط هر گره میزان تاثیر گذاری آن گره در گراف را مشخص میکند.

۴) سپس یک ماژول GNN استفاده میشود که کار انتقال پیام ها (message passing) روی G_W را انجام میدهد.

۵) آخرین نوآوری و پیشنهاد این مقاله برای تکیمل این مدل، وظیفه پیشبینی نهایی با استفاده از LM را به عهده دارد. در واقع عمل استنتاج (reasoning) روی گراف G_W در این ماژول اتفاق میوفتد. ماژول GNN بر اساس یک گراف توجه (Graph attention framework) ساخته شده است که ارائه نود ها را بر اساس یک مکانیزم رد و بدل کردن پیام بین همسایگان یک نود انجام میدهد. حال مجموع پیام های منتقل شده وارد یک MLP ۲ لایه که از batch normalization استفاده میکند، میشوند. برای هر نود t عضو گره های گراف worker از یک تبدیل خطی (Linear transformation) استفاده میکند تا از حالت اولیه به فضای R^D نگاشت شود. با مطالعه سورس کد مدل سازی این ماژول به تابعی با نام GATConvE برخورد میکنیم که سه آرگومان مهم دارد. ابعاد لایه های مخفی، تعداد انواع گره ها و تعداد ارتباطات بین گره ها. هنگامی که آرگومان اول یعنی تعداد ابعاد لایه های مخفی مشخص شود و آن را با emb_dim نمایش دهیم یک شبکه MLP داریم که لایه های آن به صورت زیر طراحی شدند:

```
nn.Linear(3*emb_dim, head_count * self.dim_per_head)
nn.Linear(3*emb_dim, head_count * self.dim_per_head)
nn.Linear(2*emb_dim, head_count * self.dim_per_head)
```

و همینطور مشخص است که از تابع فعال سازی ReLU برای این معماری استفاده شده است. در قسمت 4.3 مقاله ذکر شده است تعداد ابعاد (D = 200) و تعداد لایه ها برای ماژول GNN ۵ است با نرخ dropout ۰.۲ در هر لایه.

بررسی نتایج:

در این مقاله از ۳ دیتاستی که به صورت پرسش و پاسخ هستند استفاده شده است. نام این دیتاست ها و شرح آن ها در زیر آمده است:

۱) CommonsenseQA: ۱۲۱۰۲ سوال ۵ گزینه ای در مورد احساس مشترک است. البته استفاده از این دیتاست چالش هایی دارد از جمله اینکه test set این دیتاست به صورت عمومی در دسترس نیس.

۲) OpenBookQA: ۵۹۵۷ سوال ۴ گزینه ای که استنتاج سوالات آن نیاز به دانش علمی پایه ای دارد.

۳) MedQA-USMLE: ۱۲۷۲۳ سوال ۴ گزینه ای که به دانش پزشکی و دارویی نیاز دارد.

همانطور که در شرح دیتاست ها گفته شد، کار کردن با آنها نیاز به دانشی از پیش آموزش دیده دارد که در این مقاله برای دو دیتاست اول از ConceptNet استفاده شده است که یک گراف اصطلاحا general-domain یا عمومی است. و برای دیتاست سوم که تخصصی تر است از یک گراف ساخته شده از اطلاعات بیماری ها و بانک های دارویی استفاده شده است. در ادامه در جداول ۲، ۴ و ۵ دقت الگوریتم های مختلف روی این ۳ دیتاست اندازه گیری شده است که نتایج نشان میدهد مدل پیشنهادی

روی دو دیتاست عمومی، دقت بهتری نسبت به بقیه دارند اما در دیتاست پزشکی در رده ۴ ام الگوریتم های مقایسه شده قرار دارد.

Methods	Test
Careful Selection (Banerjee et al., 2019)	72.0
AristoRoBERTa	77.8
KF + SIR (Banerjee and Baral, 2020)	80.0
AristoRoBERTa + PG (Wang et al., 2020b)	80.2
AristoRoBERTa + MHGRN (Feng et al., 2020)	80.6
Albert + KB	81.0
T5* (Raffel et al., 2020)	83.2
UnifiedQA* (Khashabi et al., 2020)	**87.2**
AristoRoBERTa + QA-GNN (**Ours**)	82.8

Methods	Test
RoBERTa (Liu et al., 2019)	72.1
RoBERTa+FreeLB (Zhu et al., 2020) (ensemble)	73.1
RoBERTa+HyKAS (Ma et al., 2019)	73.2
RoBERTa+KE (ensemble)	73.3
RoBERTa+KEDGN (ensemble)	74.4
XLNet+GraphReason (Lv et al., 2020)	75.3
RoBERTa+MHGRN (Feng et al., 2020)	75.4
Albert+PG (Wang et al., 2020b)	75.6
Albert (Lan et al., 2020) (ensemble)	76.5
UnifiedQA* (Khashabi et al., 2020)	**79.1**
RoBERTa + QA-GNN (**Ours**)	76.1

Methods	IHdev-Acc. (%)	IHtest-Acc. (%)
RoBERTa-large (w/o KG)	73.07 (±0.45)	68.69 (±0.56)
+ RGCN (Schlichtkrull et al., 2018)	72.69 (±0.19)	68.41 (±0.66)
+ GconAttn (Wang et al., 2019a)	72.61(±0.39)	68.59 (±0.96)
+ KagNet (Lin et al., 2019)	73.47 (±0.22)	69.01 (±0.76)
+ RN (Santoro et al., 2017)	74.57 (±0.91)	69.08 (±0.21)
+ MHGRN (Feng et al., 2020)	74.45 (±0.10)	71.11 (±0.81)
+ QA-GNN (**Ours**)	**76.54** (±0.21)	**73.41** (±0.92)

و در ادامه مقایسات حالات مختلف پیاده سازی از جمله نوع ارتباط گراف، استفاده کردن یا نکردن از relevance score، انواع گراف های attention و تعداد لایه های GNN مورد بررسی قرار گرفته است.

Graph Connection (§3.1)	Dev Acc.
No edge between Z and KG nodes	74.81
Connect Z to all KG nodes	76.38
Connect Z to QA entity nodes (**final**)	**76.54**

Relevance scoring (§3.2)	Dev Acc.
Nothing	75.56
w/ contextual embedding	76.31
w/ relevance score (**final**)	**76.54**
w/ both	76.52

GNN Attention & Message (§3.3)	Dev Acc.
Node type, relation, score-aware (**final**)	**76.54**
- type-aware	75.41
- relation-aware	75.61
- score-aware	75.56

GNN Layers (§3.3)	Dev Acc.
$L = 3$	75.53
$L = 4$	76.34
$L = 5$ (**final**)	**76.54**
$L = 6$	76.21
$L = 7$	75.96

Table 7: **Ablation study** of our model components, using the CommonsenseQA IHdev set.

سوال ۲.

در این سوال قصد داریم که مفاهیم مطرح شده درباره MLP را مرور کنیم. لطفا سوالاتی که در ادامه آورده میشود را به طور کامل توضیح دهید.(۱۵ امتیاز)

- **دلیل استفاده از توابع فعالسازی در شبکه های MLP چیست؟**

وزن ها و بایاس یک معادله خطی به شکل W*x + b تشکیل میدهند. هرچند کارکردن با یک تابع خطی بسیار راحت تر است اما برای حل مسائل پیچیده نیاز است تا خروجی را از حالت خطی به حالت غیر خطی ببریم تا در حل مسئله به ما کمک کند. از طرفی مقدار خروجی W*x + b با توجه به ورودی x مقدار میگیرد و هیچ محدودیتی در مقدار ندارد و این مورد به خصوص در شبکه های با تعداد لایه ی بیشتر مشکلات جدی پدید میاورد.

- **به نظر شما کدام یک از توابع فعالسازی توضیح داده شده، استفاده بیشتری در شبکه های عمیق دارد؟ دلیل انتخاب خود را توضیح دهید.**

با مطالعه ی مقالات و تحقیقات اخیر متوجه میشویم که تابع فعال سازی ReLU (Rectified Linear Unit) پر استفاده ترین تابع فعال سازی در مدل های شبکه های عصبی عمیق است. مهمترین مزیت این تابع در کنار سادگی آن، عدم فعال سازی تمام نرون ها به صورت همزمان است که همین اتفاق باعث میشود بتوانیم شبکه هایی با تعداد پارامترها و نرون های بسیار زیاد را با توان محاسباتی کمتری آموزش دهیم.

- **مشکل ناپدید شدن گرادیان و انفجار گرادیان (vanishing and exploding gradient) را توضیح دهید و بیان کنید که در صورت استفاده از کدام یک از توابع فعالسازی احتمال رخداد این مشکلات وجود دارد؟**

محو شدگی گرادیان ، مشکلی است که حین شبکه های عصبی مصنوعی با استفاده از روش های یادگیری مبتنی بر گرادیان رخ میدهد. در این نوع روشها به منظور بروزرسانی پارامترهای شبکه عصبی از گرادیان استفاده میشود. مشکل محو شدگی اشاره به این مساله دارد مقادیر گرادیان ها با حرکت به سمت ابتدای شبکه رفته رفته به حدی کوچک میشوند که تغییرات وزن بصورت ناچیزی صورت میگیرد و به این علت فرایند آموزش بشدت کند میشود و در حالات شدیدتر این مساله باعث متوقف شدن فرایند آموزش میگردد. ناپدید شدن گرادیان در حالتی که در لایه های مخفی از توابع فعالسازی Sigmoid و یا Tanh استفاده کنیم رایج است.

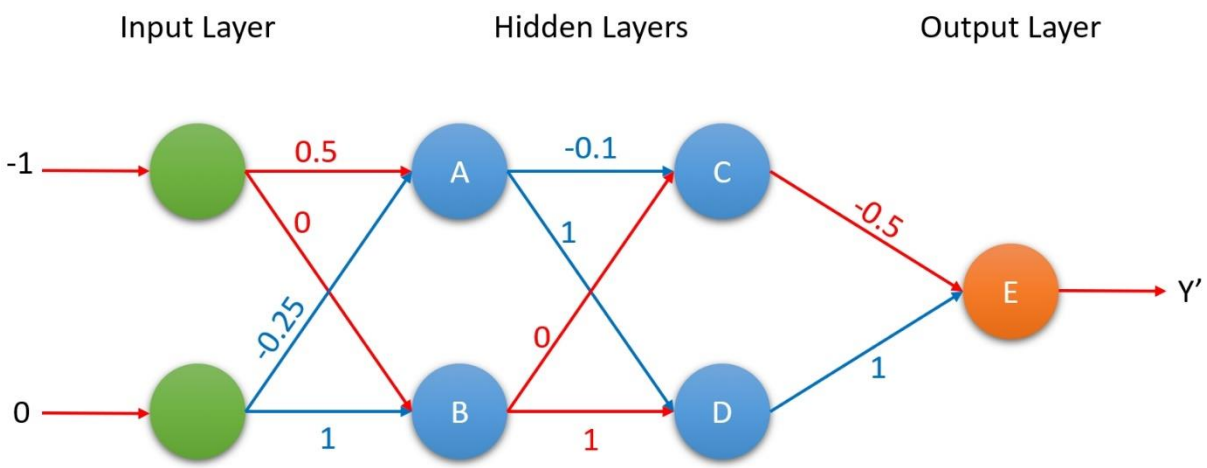
زمانی که گرادیان ها اصطلاحا منفجر میشوند، بدین معناست که مقدار گرادیان ها بسیار بزرگ میشوند و loss مدل بسیار زیاد است. در دو حالت میتوان احتمال داد که با مشکل انفجار گرادیان مواجه شدیم. مورد اول گرفتن خطاهایی مانند سرریز محاسبات است که بخاطر زیاد بودن(و بعضا بینهایت شدن مقدار گرادیان) رخ میدهد. حالت دوم زمانی است که شاهد تناوب های نامنظم در مقادیر loss هستیم. به طور معمول انفجار گرادیان به دلیل استفاده از تابع فعالسازی خاصی نیست.

- **مزیت عمیقتر کردن شبکه های MLP چیست؟آیا همیشه شبکه های با عمق زیاد نتایج بهتری نسبت به شبکه های با عمق کم تولید میکنند؟**

از مزایای عمیق تر کردن شبکه های عصبی میتوان به این موضوع اشاره کرد که توانایی یادگیری مدل افزایش میابد که بدین معناست که میتواند مسائل پیچیده تری را یادبگیرد و با دقت بهتری پیشبینی کند. اما از سوی دیگر همین افزایش قابلیت یادگیری که با افزایش عمق MLP همراه است میتواند پیچیدگی بیش از حدی به مدل بدهد که باعث افزایش مدت زمان training و همچنین مشکل overfitting روی مسائل ساده تر را داراست. پس برای قسمت دوم سوال میتوان اظهار کرد که با افزایش عمق MLP در بعضی موارد عملکرد مدل بهبود نمیابد و میتواند ثابت و یا افت داشته باشد.

سوال ۳.

شبکه ی زیر را در نظر بگیرید. در تصویر بالا ورودی ها و همچنین وزنها داده شده است. مقدار خروجی(y') را محاسبه کنید. برای لایه های مخفی مقدار بایاس را 0.25 و برای لایه خروجی 0.5 در نظر بگیرید. همچنین برای لایه های مخفی و لایه ی خروجی از تابع فعالسازی ReLU استفاده کنید. اگر مقدار واقعی خروجی (y) برابر با یک باشد با استفاده از تابع MSE خطای شبکه را محاسبه کنید.(۱۵ امتیاز)



First hidden layer:

A:

$$Output = ReLU(I_1 w_{11} + I_2 w_{21} + b)$$

$$Output = ReLU(-1 \times 0.5 + 0 \times (-0.25) + 0.25) = ReLU(-0.25) = 0$$

B:

$$Output = ReLU(I_1 w_{12} + I_2 w_{22} + b)$$

$$Output = ReLU(-1 \times 0 + 0 \times 1 + 0.25) = ReLU(0.25) = 0.25$$

C:

$$Output = ReLU(A w_{11} + B w_{21} + b)$$

$$Output = ReLU(0 \times (-0.1) + 0.25 \times 0 + 0.25) = ReLU(0.25) = 0.25$$

D:

$$Output = ReLU(A w_{12} + B w_{22} + b)$$

$$Output = ReLU(0 \times 1 + 0.25 \times 1 + 0.25) = ReLU(0.5) = 0.5$$

Y':

$$Output = ReLU(Cw_1 + Dw_2 + b)$$

$$Output = ReLU(0.25 \times (-0.5) + 0.5 \times 1 + 0.5) = ReLU(0.875) = 0.875$$

MSE:

$$MSE = (Y - \hat{Y})^2$$

$$MSE = (1 - 0.875)^2$$

$$MSE = 0.015625$$

سوال ۴.

۴ – الف)

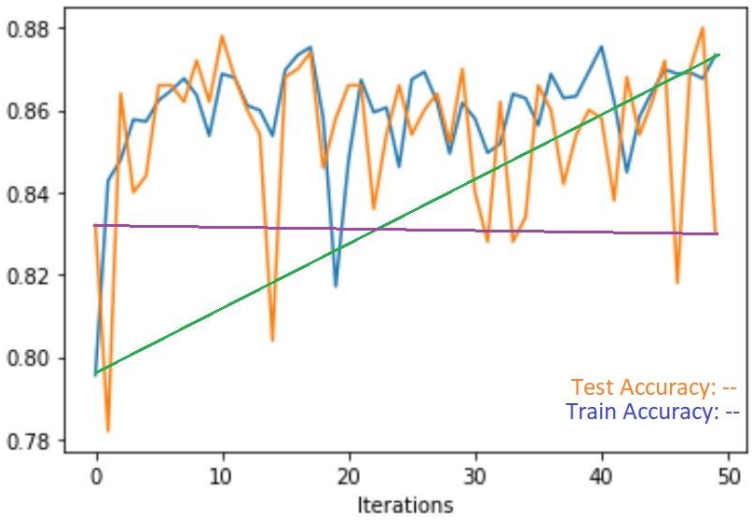
در نوت بوک داده شده، موارد خواسته شده را پیاده سازی کنید. در این نوت بوک هدف آموزش یک شبکه ساده MLP برای یادگیری مجموعه داده Fashion MNIST است. ابتدا نیاز است که مدل خود را تعریف کنید. تعداد لایه ها و تعداد نورون های هر لایه بر عهده شماست. برای تابع ضرر از Cross entropy و برای بهینه ساز از Adam استفاده کنید. سپس قسمت آموزش مدل را تکمیل کنید. در قسمت تست نیز خروجی مدل برای چند عکس موجود در داده های تست را به دست آورید و با برچسب واقعی مقایسه کنید.

نوت بوک کامل شده ضمیمه تمرین شده است. مدل پیشنهادی برای این سوال یک MLP با ۳ لایه ی مخفی که تعداد نرون ها در هر لایه مخفی به ترتیب 256، 128 و 64 است. که دقتی نزدیک به 90% دارد.

۴ – ب)

مدل تعریف شده در قسمت الف را تغییر دهید تا شبکه شما overfit شود. توضیح دهید که چطور به این نتیجه رسیدید که شبکه شما overfit شده است؟

برای overfit کردن مدل دو تغییر در مدل قبلی ایجاد کردیم. تغییر اول افزایش ظرفیت یادگیری مدل است که از یک مدل با ۴ لایه مخفی که در هر لایه به ترتیب 128، 128، 128 و 64 نرون وجود دارد. تغییر دوم در تعداد epoch هاست که به جای epoch 10 از epoch 50 استفاده کردیم. بعد از انجام عملیات training نمودار دقت مدل بر روی دیتای آموزشی و آزمون را رسم کردیم که نتیجه را در شکل زیر میبینیم. همانطور که میدانیم در دو صورت میتوان گفت مدل overfit شده است. حالت اول هنگامی است که دقت training بسیار بالا باشد (loss بسیار کم نزدیک صفر) و دقت مدل روی دیتای test با تفاوت زیادی کمتر از دقت train باشد(loss روی دیتای test بالا باشد.). و حالت دوم نیز در هنگامی رخ میدهد که دقت روی دیتای train در طول آموزش بالا رود ولی دقت روی test پس از نوسانات زیاد در انتها نسبت به اول کار آموزش تغییر محسوسی نکرده باشد. شکل زیر که خروجی ماست حالت دوم را تداعی میکند.



۴- پ)

جز موارد پیاده سازی شده در قسمت الف، توابع فعالسازی و توابع ضرر دیگر را پیاده سازی کنیدو نتایج را مقایسه کنید.(۴۰ امتیاز)

برای انجام مقایسه، به جز مدل اصلی، ۳ مدل دیگر نیز آموزش دادیم و نتایج را در جدول زیر آورده ایم. همانطور که در صورت سوال ذکر شده از توابع فعال ساز مختلف مانند ReLU، LeakyReLU و Sigmoid استفاده کردیم. همچنین در یک مورد نوع بهینه ساز را تغییر دادیم و از SGD به جای Adam استفاده کردیم.

Result Table	Train Accuracy	Test Accuracy
optimizer: Adam activation: ReLU	0.9080	**0.8780**
optimizer: Adam activation: LeakyReLU	0.8940	0.8500
optimizer: Adam activation: Sigmoid	0.8500	0.8420
optimizer: SGD activation: ReLU	**0.9100**	0.8760

References

1) https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253

2) https://www.analyticsvidhya.com/blog/2020/01/fundamentals-deep-learning-activation-functions-when-to-use-them/

3) https://programmathically.com/understanding-the-exploding-and-vanishing-gradients-problem/

4) https://thedatafrog.com/en/articles/overfitting-illustrated/

5) https://towardsdatascience.com/understanding-pytorch-activation-functions-the-maths-and-algorithms-part-1-7d8ade494cee

6) https://machinelearningmastery.com/how-to-choose-loss-functions-when-training-deep-learning-neural-networks/

7) https://deeplearninguniversity.com/pytorch/pytorch-activation-functions/